Entity Retrieval over Structured Data

Hui Fang, Rishi R. Sinha, Wensheng Wu, AnHai Doan, ChengXiang Zhai
University of Illinois, Urbana-Champaign, IL, USA
{hfang,rsinha,wwu2,anhai,czhai}@cs.uiuc.edu

Abstract

Entity retrieval is the problem of finding information about a given real-world entity (e.g., director Peter Jackson) from one or a set of data sources. This problem is fundamental in numerous data management settings, but has received little attention. We define the general entity retrieval problem, then discuss the limitations of current information systems (e.g., relational databases, search engines) in solving it. Next, we focus on the specific problem of entity retrieval over structured data (as opposed to text or Web pages). We show that it is inherently more general and difficult than the actively-studied problem of entity matching (i.e., record linkage).

We then develop the ENRICH *system, which significantly extends entity matching solutions to perform entity retrieval. In particular,* ENRICH *employs clustering techniques to obtain a global picture on how many entities are "out there" and which data fragment should best be assigned to which entity. It also constructs profiles that capture important characteristics of the target entity, then uses the profiles to help the assignment process. Finally, it leverages "query expansion", an idea commonly used in the information retrieval community, to further improve retrieval accuracy. We apply* ENRICH *to several real-world domains, and show that it can perform entity retrieval with high accuracy.*

1. Introduction

We use the term "entity retrieval" (or "ER" for short) to refer to the problem of finding information about a *given real-world entity* from a set of data sources. For example, we may want to find all movies and reviews of a director named Peter Jackson from several online movie databases. As another (simplified) example, given the database of three research papers shown in Figure 1, we may want to find all papers written by a researcher named "Chris Zhai", who has an ICDE-01 paper called "Query Optimization". The result in this case is papers (1)-(2).

Entity retrieval *mines information* about a given entity from the data, and thus plays a fundamental role in numerous information processing contexts. Companies often want to mine information about their competitors or products from online discussion groups. Bioinformatists often must retrieve all information related to a gene, from structured data and text (e.g., medical literature). Intelligence analysts frequently must retrieve information about a person or group, given some rudimentary initial information. One of the authors bought a house recently, and found himself spending countless hours with Google, trying to retrieve all information about a particular person (e.g., realtor, home inspector) or organization (e.g., mortgage lenders).

Query: Find papers of X, given (a) X's name is Chris Zhai, and (b) X has an ICDE-01 paper called "Query Optimization".

Database: (1) C. Zhai, B. Croft, "Passage Retrieval", KDD-00
(2) Chris Zhai, "Data warehousing", SIGMOD-99
(3) A. Doan, A. Kramnik, "Semantic Web", WWW-02

Figure 1. An example of entity retrieval from a publication database.

Despite its obvious importance, to date ER has received very little attention (see the Section 5). In this paper we take an initial step in addressing this situation. We begin by defining the general ER problem, and discussing the limitations of current information systems in solving it. Next, we focus on ER over structured data (e.g., retrieving all papers of Chris Zhai, as described above), leaving the handling of text and Web data as future research.

To attack ER over structured data, we transform all relevant data fragments (e.g., papers in Figure 1), as well as the ER query, into *relational tuples*, then match the tuples to find all those that refer to the same real-world entity as the query tuple. As such, our ER setting is quite related to entity matching, a problem which has been studied extensively under the names "record linkage" or "tuple deduplication" [13, 24].

However, ER is fundamentally more difficult than entity matching for the following reason. Entity matching typically assumes the tuples to be matched describes the *same aspects* of the target entity. For example, the two tuples (Mike Smith, (217) 344 2583, Illinois) and (M. Smith, 344 2583, Champaign IL) both describe names, phone numbers, and addresses. In such cases, matching can be de-

cided fairly accurately by examining the similarity of these many same aspects.

In ER contexts, on the other hand, tuples often describe *very different aspects* of the same entity, and often are related only via some form of *names*. For example, in the database of Figure 1, each paper describes a different physical publication at different time points. They can have very different sets of co-authors, titles, and conferences. Thus, in deciding if two papers, say, (1)-(2) in Figure 1 are written by the same author, we can rely mostly only on matching the names "C. Zhai" and "Chris Zhai". However, matching based on the names alone is not reliable, and in general, applying entity matching techniques directly to our ER context does not yield high accuracy, as we confirm empirically in Section 4.

In this paper we describe the ENRICH system that significantly extends entity matching techniques to perform ER efficiently. The key innovations that we make are the following:

Global Matching with Clustering: Since pairwise matching of tuples is unreliable, we obtain more global knowledge to help the matching process. Specifically, we cluster tuples to obtain a rough picture of the entities "out there", then use this picture to assign tuples to entities. For example, given a paper A written by C. Zhai in SIGIR-02 and a paper B given by Chris Zhai in SIGMOD-01, considering them in isolation we may decide that they match (i.e., refer to the same person), since the names are similar, and it is conceivable that the same person publishes in both SIGIR and SIGMOD. However, if the clustering process tells us that there is a researcher named Cheng Zhai with many papers in SIGIR, and another researcher named Christopher Zhai with many papers in SIGMOD, then it is more likely that paper A should be assigned to the former researcher, and paper B to the latter. At the end, each cluster of tuples is assumed to represent a single entity, and the cluster that best matches our initial knowledge of the target entity is returned as the ER result.

Sanity Check using Profilers: The result cluster can still contain tuples that do not belong to the target entity. For example, the cluster for the Chris Zhai query may contain four SIGIR-02 papers and three SIGMOD-02 papers. Our knowledge of IR and database researchers may suggest that having seven such major papers in the same year is highly unlikely, and that the result cluster is still incorrect. We develop a novel approach that exploits this intuition to improve ER accuracy. Specifically, we construct a set of modules called *profilers*, each of which captures *some characteristics of a typical instance* of the target entity (e.g., database researcher). A sample profiler in the database domain may state that a database researcher is unlikely to have seven papers in a top conference in a year. Given the profilers, we iteratively removes tuples from the result cluster until it best satisfies the constraints imposed by the profilers. It is important to emphasize that the key to the effective use of profilers is the fact that we can *merge tuples* to obtain *aggregate* properties of the target entity (e.g., publication history, average movie rating, etc.). Profilers are then applied to both aggregate and non-aggregate properties, to "sanity check" the ER result.

Tuple Expansion: To further improve ER accuracy, we "enrich" each tuple in the data, as well as the query tuple, with information from "neighboring" tuples. The intuition is that the closest neighbors (in semantic distance) are likely to refer to the same real-world entity, and hence can "lend" some correct information. The more information we have in a tuple, the more accurately we can match it with others. This idea is reminiscent of the query expansion idea in information retrieval [23, 6], except that here we expand both the query and data tuples, and that the context we are considering is different.

In summary, this paper makes the following contributions:

- We formally introduce entity retrieval. As far as we know, this is the first paper that examines this important problem in depth.
- We develop a solution for entity retrieval over structured data, as embodied by the ENRICH system. The solution innovates by exploiting global knowledge using clustering, and merging retrieved tuples to check for the validity of aggregate properties using profilers. It further adapts the idea of query expansion in IR, to improve ER accuracy.
- We apply ENRICH to several real-world domains, and show that it can achieve high retrieval accuracy, with 67-95% precision and 78-93% recall. The experiments also demonstrate the utility of our solution components.

The paper is organized as follows. Section 2 defines the entity retrieval problem. Section 3 describes the ENRICH system. Section 4 presents the experiment results. Section 5 reviews related work. Section 6 discusses future work and concludes.

2. The Entity Retrieval Problem

In this section we first define a conceptual framework for ER. Next, we discuss the limitations of current information systems in solving this problem. Finally, we describe the specific ER problem to be addressed in the paper.

2.1. A Conceptual Framework for ER

We assume the user models the domain of interest with a conceptual schema S, which consists of entity sets, relations, and attributes. Figure 2 shows a simple schema for the

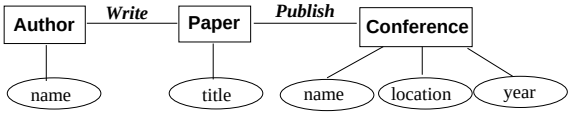


Figure 2. A sample conceptual schema for the publication domain.

(C) IF $\exists e$ such that Author(e) AND e.name contains "Clinton"
AND ($\exists p, c$ such that
Write(e,p) AND Publish(p,c) AND c.year=2003)

(R) THEN $\forall x, y$ such that Write(e,x) AND Publish(x,y)
Return tuple (x.title, y.year)

Figure 3. An ER query over schema S of Figure 2.

publication domain, represented as an entity-relationship diagram. The schema has three entity sets, Author, Paper, and Conference, and two relations, Write and Publish. Conference, for example, has three attributes name, location, and year.

The conceptual schema S models real-world entities that the user may be interested. An ER query Q retrieves information about such an entity. Formally, it is a tuple (C, R), where constraint expression C specifies the type of the target entity (e.g., Person or Paper), and what the user knows about it, and template R specifies what the user wants to retrieve about the entity, and in which format.

Figure 3 shows a sample ER query expressed in a pseudo-English format, for ease of readability. Constraint C specifies an author e, whose name contains the word "Clinton" and who has a paper published in 2003. Template R states that the user wants to obtain all papers and years of publication of this author, as a list of tuple (paper,year). In general, an ER query can be expressed using a logic or database query language, such as XQuery or SQL.

Given a dataset D, evaluating an ER query Q conceptually proceeds in the following steps:

1. Extract data fragments (from D) that could be relevant to answer Q. Use them to form a data graph G that conforms to domain schema S.
2. Evaluate constraint C over data graph G, to identify a set I of nodes that refer to entities that the user may be interested in. Then ask the user to select a node from I that refers to the entity of interest. This node becomes the "seed" node.
3. Examine graph G to find other nodes that refer to the same entity as the seed node.
4. Evaluate template R over all nodes found in Step 3, and return the results.

Example 1 *To evaluate ER query Q in Figure 3 over dataset D in Figure 4, Step 1 converts D into data*

(1) William Clinton, Hillary Clinton, I feel your pain, SIGWASH, Seattle, WA, 2003.
(2) H. R. Clinton, It takes a town, SIGMOD, Baltimore, MD, 2001.
(3) G. Bush, H. Clinton, Compassionate Conservatism, SIGRICH, Champaign, IL, 2004.

Figure 4. A sample publication dataset

.

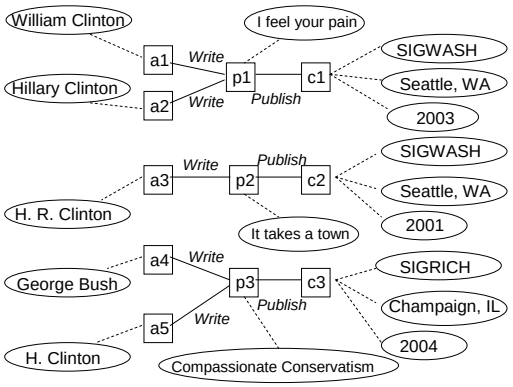


Figure 5. A data graph of the dataset in Figure 4

.

graph G in Figure 5 (that conforms to schema S of Figure 2). Each square node of E represents an entity, each solid edge a relationship, and each dotted edge an attribute.

Step 2 evaluates the constraint "the person name contains Clinton and that he or she published in 2003" over G and produces nodes a_1 and a_2. These nodes represent William Clinton and Hillary Clinton, both of whom published in 2003. Suppose the user then selects node a_2. a_2 becomes the "seed" node, and the real-world entity that the user is interested in is Hillary Clinton.

Step 3 finds all person nodes that match, that is, refer to the same real-world person as node a_2. On data graph G (Figure 5) these are nodes a_3 and a_5.

Step 4 evaluates template R of the query (Figure 3) over nodes a_2, a_3, and a_5 to produce the following list of the tuples:

(I feel your pain, 2003)
(It takes a town, 2001)
(Compassionate Conservatism,2004)

Step 2 is necessary because in practice users often underspecify a target entity, because of incomplete knowledge, impatience, unfamiliarity with the data set, or the limitation of the query interface. Evaluating the ambiguous constraint C over a data graph then often produces multiple matching nodes, which can refer to multiple entities. As a result, users must intervene and "pintpoint" a node which refers to a single target entity, before the ER process can continue.

Step 3 reconciles the entities represented by the nodes of a data graph, and is at the heart of this paper. It is necessary because real-world data sets are often noisy, with no correct entity identification mechanism. Hence, multiple nodes in a

data graph can refer to the same entity (e.g., nodes a_2, a_3, and a_5 as discussed above).

2.2. ER in Current Information Systems

Many current types of information systems already perform some variants of the above ER process, albeit in very limited fashions. Relational database systems can perform ER if entities can be cleary distinguished using some attribute combinations as a key. For example, if the authors can be distinguished using their names or an ID attribute, then the ER query Q in Figure 3 can be written as a SQL query, and the ER process reduces to evaluating this SQL query over the relational data.

However, such systems fail to perform ER in the common case of noisy data (e.g., the DBLP database [1]), with no entity identifying attributes. Indeed, when executing ER query Q of Figure 3 as a SQL query on data graph G of Figure 5 (which contains publications in DBLP format), a relational database engine will only perform *syntactic matching* and return publications of both William Clinton (node a_1) and Hillary Clinton (nodes a_2, a_3, and a_5), a clearly undesired outcome.

Many Deep Web databases (those behind a query interface) perform ER, but either *manually*, or in some automatic, but simple and inadequate fashion. The Internet Movie Database (IMDB at $imdb.com$), for example, takes a user query such as "Helen Hunt" and return a list of possible target entities: Helen Hunt actress I, Helen Hunt actress II, Helen Hunt makeup artist, etc. Once the user has clicked on a name, say, Helen Hunt actress I, the site returns all movie-related information about that person. However, this information is manually gleaned from the data sources of IMDB, a costly and labor-intensive undertaking.

The DBLP site [1] also takes a partial author name (e.g., Gupta) and produces a list of author names. Once a name has been selected, however, in most cases DBLP simply returns all papers which contain that *exact name*. This *syntactic matching* on author names is clearly inadequate, since the same name (e.g., Ying Lu) in many cases refer to different authors. Other examples of Deep Web databases that perform some ER variants include comparison shopping sites and product stores such as $amazon.com$.

In a sense, current search engines also perform some limited ER variant. For example, given query "George Bush", the top ten URLs Google produced mention both the current president and the elder Bush. Clicking on the "similar pages" button next to each URL then retrieves more information about the topics of that URL, including more information about the person or persons mentioned in the URL. However, this does not work well in many cases as an ER functionality, because the notion of similarity here is not defined based on any particular entity.

In sum, it is clear that entity retrieval is an important and practical problem that has not been addressed well by research and practice. This paper takes a first step in addressing this problem. In what follows we define the specific ER problem that is the focus of the paper.

2.3. ER over Structured Data

As the first step, we consider the problem of performing ER over structured data (e.g., relational or XML). This problem very commonly arises in practice (e.g., retrieving all papers by an author from DBLP, as described earlier). We leave ER over text or combination of text and structured data as a topic of future research.

We assume that the structured data D conforms to a schema T. We also assume that semantic mappings between the elements of T and the user schema S have been created (either manually or semi-automatically using techniques such as [21]). The mappings allow us to convert D into a data graph that conforms to schema S, thereby solving Step 1 of the ER evaluation process.

Since executing Step 4 is straightforward, in the rest of the paper we focus on Steps 2 and 3, with a particular emphasis on Step 3. In the next section, we describe our ENRICH solution that addresses these two steps.

3. The ENRICH Solution

Throughout this section, we will motivate our solution using the DBLP-Lite data set shown in Figure 6, and the ER query that finds all papers written by an author named Chris C. Zhai. The result is papers (1)-(3). The ENRICH solution implements the conceptual ER evaluation process outlined earlier as follows.

3.1. Preprocessing

Find Relevant Data Nodes: Recall that Step 2 of the ER evaluation process (Section 2) evaluates constraint C on the data graph to find the set I of all nodes that could refer to the target entity. From now on, we will refer to I as the "umbrella set". Clearly, we want this set to be as *complete* as possible, to avoid missing the target entity, but as *small* as possible, to reduce subsequent processing costs and to not overwhelm the user in the selection step. We believe achieving these two requirements is best left to specific applications, since each specific type of entity may require a different kind of algorithm for evaluating constraint C.

For example, in the publication domain, to find the "umbrella set" for the query in Figure 6, we can find all papers with an author name possibly matching "Chris C. Zhai". Utilizing a name matching tool [5], this would return the papers (1)-(7) (i.e., removing paper (8)). Notice that the matching tool may have special knowledge on how to match names, which takes into account the abbreviation of first and

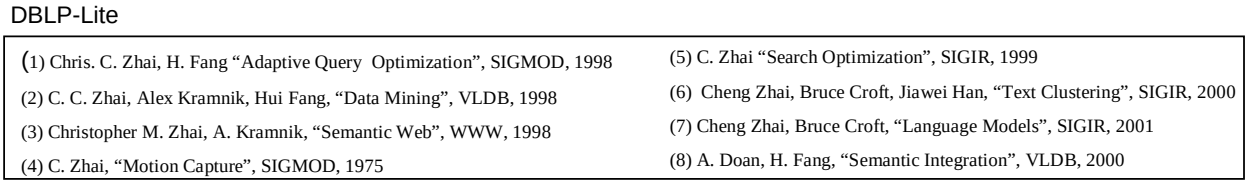


Figure 6. An example of entity integration from DBLP-Lite data source.

middle names. Such tools would perform much better than generic tools that match strings based on, say, edit distance [5].

Selecting a Seed Node: Given the umbrella set, the user will select a node that refers to the target entity. Suppose in the DBLP-Lite case that the user selects paper (1). Then this paper refers to a unique author, and the paper becomes a "seed" node. The problem then becomes matching the seed node with each node in the umbrella set.

Converting Nodes in Umbrella Set into Tuples: Next, we convert each node in the umbrella set into a relational tuple, so that we can judge the similarity of nodes using tuple-based matching methods (e.g. [13, 24]). A relational tuple consists of a set of (feature,value) pairs. Currently we select the features manually, based on the domain schema S. We select features that best capture the most salient aspects of the target entity, and that would help match entities. In the future, we plan to apply feature selection techniques to automate this step.

For example, in the DBLP-Lite domain, we select the features author name, co authors, paper title, conference name and conference year. The paper (2) in Figure 6 then becomes the tuple (C. C. Zhai, {Alex Kramnik, Hui Fang}, Data Mining, VLDB, 1998). Note that the feature co author is set-valued.

Once the conversion is done, the umbrella set is a set of tuples, with a seed tuple (which may be augmented with information from the ER query, when appropriate). Our problem then is to find all tuples in the set that refer to the same real-world entity as the seed tuple. In the rest of the current section we focus on this problem. In describing the publication example, we will use "tuple" and "paper" interchangeably when there is no ambiguity.

3.2. Pairwise Matching with Seed Tuple

A straightforward solution to the above problem is pairwise entity matching: match the "seed tuple" (e.g., tuple (1) in Figure 6) with every other tuple in the umbrella set, (e.g., tuples (2)-(7)) and retrieve all matched tuples.

If two tuples refer to the same real-world entity, they must agree on some attributes. For example, if tuples (1) and (2) refer to the same author, the author name attribute must have the same value, albeit this value can appear in syntactically different forms (e.g., "Chris Zhai" vs. "C. C. Zhai"). We refer to such attributes as *shared attributes*, and the remaining attributes in tuples as *disjoint attributes*.

The simplest pairwise matching method is to match two tuples only based on the *shared* attributes. Specifically, we can use a similarity function defined on all shared attributes and a similarity threshold to retrieve all tuples about an entity. We call this method **S**haredMatch. For example, in Figure 6, **S**haredMatch retrieves all papers X such that an author name of X matches "Chris C. Zhai", based on some name-matching criterion (e.g., [5]). This method would return papers (1)-(7), with papers (4)-(7) being false positives.

However, such a simple matching method ignores useful context information. Observe that the "seed" paper (1) contains not just the name "Chris C. Zhai", but also some "context information" surrounding this name: the co-author H. Fang, a paper title, the conference "SIGMOD", and the year 1998. Thus one way to improve **S**haredMatch is to use a more sophisticated similarity function which examines not only the shared attributes but also the disjoint ones. We call this method **A**llAttrMatch.

For the example in Figure6, the co-authors and conferences provide useful additional evidence regarding whether two papers are written by the same author. Hence **A**llAttrMatch would match two papers based on author name as well as co-author and conference similarities. As a result, papers (3), (6), (7) would be excluded because of low co-author similarity, even though the names "Cheng Zhai", "Christopher Zhai" and "Chris C. Zhai" are similar. The result is papers (1),(2),(4), (5), with paper (3) being false negative and papers (4) and (5) being false positives.

Similarity Functions: There are many ways to define a similarity function on all the attributes. We describe a simple yet general method, which uses logistic regression [14] to combine the similarities of individual attributes. Given any two tuples t_1 and t_2, we use the binary variable y to denote whether they refer to the same entity. y has a Bernoulli distribution with parameter π, where $\pi = p(y = 1) = p(Match|t_1, t_2)$ is the probability that tuples t_1 and t_2 refer to the same entity. The similarity function between tuples t_1 and t_2 can then be written as $s(t_1, t_2) = p(Match|t_1, t_2)$, which means that the similarity between two tuples is equiv-

alent to the probability of their referencing the same entity.

Formally, let $A = \{a_1, ..., a_n\}$ be the attributes of tuples t_1 and t_2, and let s_i be a similarity function defined for attribute a_i, for $i \in [1, n]$. By using logistic regression to combine the similarity measure s_i on each attribute a_i, we get

$$\log \frac{s(t_1, t_2)}{1 - s(t_1, t_2)} = \sum_{i=1}^{k} \lambda_i s_i(t_1.a_i, t_2.a_i)$$

where λ_i is the weight for s_i.

The weights λ_i can be calibrated automatically by using some training data as follows. Suppose $D = \{(t_1^1, t_2^1, y_1), ..., (t_1^n, t_2^n, y_n)\}$ is a set of training examples (i.e., known matchings), where $y_i \in \{0, 1\}$ indicates whether t_1^i and t_2^i are related to the same entity. Let $\Lambda = (\lambda_1, ..., \lambda_k)$ be all the parameters. The likelihood of the training examples is

$$p(D|\Lambda) = \prod_{i=1}^{n} p(Match|t_1^i, t_2^i)^{y_i} (1 - p(Match|t_1^i, t_2^i))^{1-y_i}$$

The similarity weights Λ can now be estimated using the maximum likelihood estimator $\Lambda^* = argmax_\Lambda p(D|\Lambda)$ [14]. Therefore, the learned **A**llAttrMatch similarity function can be written as

$$s(t_1, t_2) = p(Match|t_1, t_2) = \frac{exp(\sum_{j=1}^{k} \lambda_j s_j(t_1.a_j, t_2.a_j))}{1 + exp(\sum_{j=1}^{k} \lambda_j s_j(t_1.a_j, t_2.a_j))},$$

where $exp(x) = e^x$.

3.3. Global Matching with Clustering

The above two methods perform only *pairwise matching*: for each paper X, they match X with "seed" paper (1) by considering only the evidence in X and (1). If X is sufficiently close to (1), then X is retrieved.

A major deficiency of viewing the ER problem as *pairwise matching* is that it could not exploit *global* knowledge about the results. We claim that one can often do better by considering the "global picture", and hence all evidence. To explain, consider two papers (1) and (5). Since author names are similar ("Chris C. Zhai" vs. "C. Zhai") and the conferences are quite close ("SIGMOD" vs. "SIGIR"), a pairwise system may decide that the two papers match. However, if we examine all papers (1)-(7), we would notice that there is a person named "Cheng Zhai" who is quite active in SIGIR (see papers (6)-(7)), around that time period. Given this information, it seems more likely now that paper (5) belongs to this person, and not to person "Chris Zhai" of paper (1).

We now present a clustering method for ER, which attempts to model different entities globally so that the global context information can be exploited to improve retrieval performance for a particular entity. We propose to use an agglomerative hierarchical clustering (AHC) method [15] to cluster all tuples in the umbrella set by incrementally merging them.

Given a pairwise similarity function, AHC would first merge the two most similar tuples, forming a two-tuple cluster, which replaces the two original tuples. Then it continues merging the tuples or clusters that are most similar until the similarity between the two most similar clusters/tuples is below a threshold, when the clustering process would stop. At the end of clustering, the cluster containing the seed tuple is taken as the results.

One advantage of AHC is that it does not require a predefined number of clusters, though it does require a threshold, which can be calibrated on a training set. AHC has several variants corresponding to different ways of defining the similarity between two clusters, including single-link, complete-link, and average-link methods [15]. We explore single-link and average-link in our experiments. These methods define the similarity between two clusters as the maximum similarity and the average similarity, respectively, over all pairs of tuples in the two clusters.

Returning to the example in Figure 6, at the beginning, each paper is regarded as a single cluster. With the clustering methods, after several iterations, we would get the following clustering: {papers (1) and (2)} (i.e. Chris Zhai,SIGMOD); {paper (3)} (i.e. Christopher M. Zhai, WWW); {paper (4)} (i.e. C.Zhai, SIGGRAPH); {paper (5)} (i.e. C. Zhai, SIGIR); {papers (6) and (7)} (i.e. Cheng Zhai, SIGIR) and {paper (8)}(i.e. A.Doan, VLDB). It is clear that paper(5) more likely belongs to Cheng Zhai who is active in SIGIR. Therefore, we would further merge paper(5) with paper (6) and (7) to form a new cluster. Similarly, we would further merge other clusters until the similarity between any two clusters is below the threshold. At the end, we take the cluster containing the seed tuple as the result, which contains papers (1)(2)(4).

3.4. "Sanity Checking" with Profilers

Consider the output of the above solution: papers (1),(2),(4). If this output is correct, then we would have a researcher (named Chris C. Zhai) who was in database community in 1975. After 1975, he did not publish any paper until 24 years later. Our knowledge about a typical database researcher would suggest that such a gap in publication history is unlikely, and that the output is not correct yet.

We develop a novel approach that exploits this observation to improve ER accuracy. Specifically, we construct a set of modules called *profilers*, each of which captures *some integrity constraints* that the target entity E would be likely to satisfy. For example, a profiler can capture a constraint such as no database researcher has ever published more than five SIGMOD papers in one single year. Given a set of tuples

$(t_1, t_2, ..., t_n)$ that reference E, a profiler P issues a confidence score on taken together how well those tuples fit the "profiler". Profilers typically exploit the aggregate attributes $c_1, ..., c_k$ of a set of tuples that represent some ER results.

Given a result cluster R and a profiler f_p, our current profiling algorithm iteratively remove tuples from R, in order to find the maximum subset of R which satisfies the profiler. For any set of tuples R', R' satisfies the profiler if and only if $f_p(R')$ is larger than a given threshold θ. At each iteration, we try to remove the tuple whose removal leads to the maximum change of the f_p until we find the subset satisfying the profiler (if R already satisfies the profiler then we do not remove any tuple from R).

The threshold θ is tuned based on training examples. In general, the profiler function f_p can be any binary classifier that gives a confidence value to measure the "coherence" and consistency of a set of tuples as the retrieval result for a real world entity. A profiler can either focus on only one constraint (e.g. no researcher can publish the papers in more than two areas in one year), or combine many different component profilers, e.g., through logistic regression as done in our experiments. The combination weights can again be learned automatically based on a set of training examples. Section 4 describes using profilers in our experimental domains in detail. Profilers can also be manually constructed by users.

Continuing with the publication example, Assume we have a profiler which states that a researcher usually does not have a gap of more than twenty years for two consecutive publications. In the example of DBLP-Lite, we would remove paper(4) from the result so that the modified output best fits the profiler. Note that the key to enable this solution is the fact that we can "integrate" the papers, to obtain *aggregate* properties of researcher Chris C. Zhai (publication history in this case). Profilers are then applied to both aggregate and non-aggregate properties of the researcher.

3.5. Expanding Tuples using their Neighbors

So far we use only the information of one (seed) paper to find the author. However, one paper can not represents the whole information about the author. For example, in most cases, the co-authors of one paper do not cover all the possible co-authors. According to "seed" paper (1), Chris C. Zhai only has one co-author: H. Fang. Based on it, paper (3) would not be returned as result. However, if we consider his another paper, i.e., paper(2), then it is clear that A. Kramnik is another co-author of Chris C. Zhai, and that paper (3) should also be returned.

Thus, we leverage "query expansion", an idea commonly used in the IR community, to further improve ER accuracy. Here we assume the most relevant tuple (i.e. the tuple which is the most similar to the seed tuple) refers to the same entity as the seed tuple does. With this assumption, we could merge the original seed tuple and the most relevant tuple to form a new aggregate tuple, which contains more information about the target entity. The new aggregate tuple will be used as new "seed" tuple.

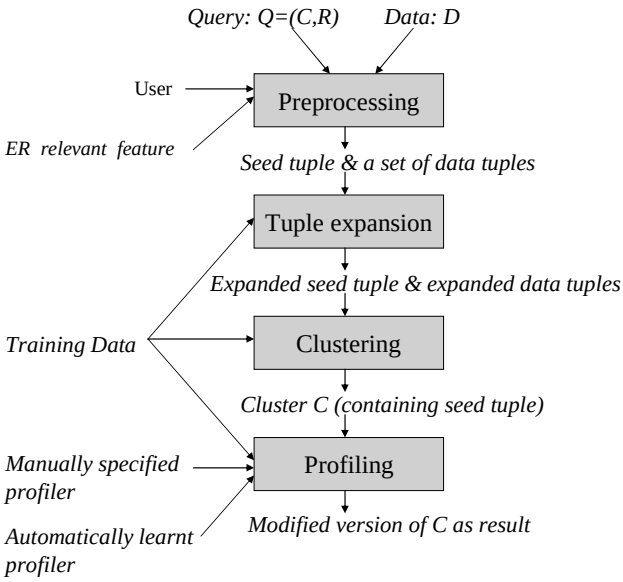


Figure 7. The ENRICH architecture

Consider again the DBLP-Lite example. The paper (2) is the paper most similar to the "seed" paper (i.e. paper (1)). We could expand paper (1) with paper (2) to form a new seed paper, which indicates that Chris C. Zhai wrote the paper with two co-authors H. Fang and Alex Kramnik and his name is also written as C. C. Zhai. With the use of the new seed paper, we would also return paper (3) as result.

Besides expanding the "seed tuple", we consider expand every tuple in data set as well. We call theses "query expansion" and "data expansion", respectively. Since it is not always the case that a tuple and its most similar tuple refer to the same entity, instead of augmenting each tuple with its most similar tuple, we augment it with all tuples whose similarity is above some threshold. The threshold is tuned based on training data. We omit further details due to space limitation.

3.6. The Final ENRICH Architecture

Figure 7 shows the main components of the ENRICH system, which employs all of the above main ideas. The preprocessing module finds the umbrella set, consults the user to select the seed data, then converts all data fragments to tuples. The tuple expansion module then performs both query and data expansions. Clustering and profiling are then carried out, as explained earlier.

Before tuple expansion, clustering, and profiling, ENRICH uses a training dataset to learn system parameters (e.g., weights in similarity functions and coefficient for pro-

filers) and select thresholds (e.g., stopping threshold for clustering).

4. Empirical Evaluation

We have experimented with ENRICH on several real-world datasets. Our goals are to evaluate the ER accuracy and to examine the usefulness of different system components.

Data Sets: The first data set was taken from DBLP ($www.acm.org/sigmod/dblp$). Here the ER problem is to find all the papers written by a given author whom we specify through a seed paper tuple. The second data set was taken from IMDB ($www.imdb.com$). Here we find all movies played by a given actor whom we specify through a seed movie tuple.

Both data sets were generated as follows. (1) Select some ambiguous seed names (i.e., author names in DBLP and actor names in IMDB). We focus on ambiguous names because the ER task is more challenging in these cases. (2) Augment each seed name with about 10 variants obtained through searching for the most similar names in the corresponding database. The idea is to capture variations such as "D. Smith" and "David Smith". (3) Download all tuples that syntactically match any of these names. For DBLP, we download all papers written by any of our candidate names, and record the author names, title, publication conference and publication year. For IMDB, we download all movies played by any of our candidate names, and record the director, writer, genre, cast, year and role.

The DBLP set has 1837 papers and 327 distinct author names, the IMDB set has 3473 movies and 300 distinct actor names. All the tuples obtained according to the same seed name are referred to as an "umbrella set". Both data sets have 30 umbrella sets each. We randomly split each data set into a training set with 10 umbrellas and a test set with 20 umbrellas. For each umbrella set, we randomly choose 3 tuples as seed tuples to define 3 distinct ER tasks. We label all the tuples with true real world entities, and treat these judgments as our gold standard.

Performance Measure: We evaluate ER results using precision P (i.e., the percent of tuples retrieved that are truly about the target entity), recall R (i.e., the percent of all the tuples about the target entity that are retrieved), and $F1 = \frac{2PR}{P+R}$. The performance of a data set is computed by averaging P, R, and $F1$ over all the seed tuples in the set.

Similarity Functions: In both data sets, the only shared attribute is a person's name. Hence, the **S**haredMatch method measures the similarity between two names, which we follow [5] and define as the Jaccard coefficient of the corresponding tri-grams of two names. The **A**llAttrMatch method uses additional attributes when matching two tuples. For DBLP, we used co-author name and type of conference. For IMDB, we used role of a person in the movie, genre of the movie, and production year.

The overall **A**llAttrMatch similarity function is computed by using logistic regression to combine the similarity functions on each attribute, as described earlier. This yields the similarity function for DBLP:

$$s(t_1, t_2) = \frac{exp(4.9s_n + 8.5s_{ca} + 2.4s_c - 4.2)}{1 + exp(4.9s_n + 8.5s_{ca} + 2.4s_c - 4.2)}$$

where $exp(x) = e^x$ and s_n is the name similarity. The co-author similarity s_{ca} indicates whether the two papers share a co-author. The conference similarity s_c is based on manually classified conference categories.

The learned similarity function for IMDB is:

$$s(t_1, t_2) = \frac{exp(15.3s_n + 0.21s_g + 1.6s_r + 1.6s_y - 15.8)}{1 + exp(15.3s_n + 0.21s_g + 1.6s_r + 1.6s_y - 15.8)}$$

where $exp(x) = e^x$ and s_n is the name similarity. The genre similarity s_g indicates whether the two movies have an overlapping genre. The role similarity s_r indicates whether the person played the same role in the two movies. The year similarity s_y indicates whether the production years of the two movies are within 10 years.

Profilers: In both data sets, we combine multiple profilers to obtain the best ER performance. The profilers for DBLP focus on the aggregate characteristics of the publication activity of an author. We choose the following four aggregate characteristics: (1) the number of distinct types of conference; (2) the largest publication gap;(3) the largest publication ratio; and (4) the entropy of publication distribution. Note that the purpose of the features (2) and (3) is to lower the confidence when the publication distribution is uneven, and the idea of (4) is to check with the overall performance on the publication distribution over the years for a typical author.

We then build a single profiler that combines these four characteristics. We use logistic regression on a training set to learn how to combine the characteristics. We omit the detail of the learning process for space reasons. The final learned profiler is

$$P(R) = 3.7 - 0.59NC - 0.3LPG - 1.1NMP - 0.3EP$$

where NC is the number of distinct conferences, LPG is the largest publication gap, NMP is the normalized maximum number of publications, and EP is the entropy of publication distribution defined by $EP(R) = -\sum_i p(y_i) \log p(y_i)$, where y_i is a year, and $p(y_i)$ is the empirical distribution of number of publications in year y_i.

The learned profiler for IMDB is based on similar aggregate attributes, and is

$$P(R) = 3.2 - 0.93NR - 0.08LYG - 1.3NMM - 0.3EM$$

where NR is the number of distinct roles, LYG is the largest year gap, NMM is the normalized maximum number of movies per year, and EM is the entropy of movie distribution defined similarly to the entropy of publication distribution EP.

Experiments: For each data sets, we perform four sets of experiments. First, we measure ENRICH's accuracy, and compare its performance with baseline methods. Second, we conduct lesion studies to measure the contribution of each component in the system to the overall performance. Third, we analyze under what situation clustering would help to improve the performance and conduct the corresponding experiment to test our analysis. We also study the difference between the two clustering methods (i.e. single-link and average-link). Fourth, we conduct sensitivity analysis for the main parameters of our system. Finally, we apply ENRICH to detect the problematic DBLP homepages of hundreds of database researchers.

4.1. Retrieval Accuracy

Table 1 shows the average accuracy of different ENRICH variations. **SC** and **AC** refer to single-link and average-link clustering, respectively. Prof refers to profiling, while **QE** and **DE** refer to query expansion and data expansion, respectively. A method such as **SC + Prof** means we do only single-link clustering followed by profiling (no query and data expansion).

Table 1. Retrieval Accuracy of Different Methods

Method	DBLP			IMDB		
	F1	Prec	Rec	F1	Prec	Rec
SharedMatch	0.61	0.59	0.85	0.56	0.46	1.0
AllAttrMatch	0.65	0.60	0.90	0.75	0.95	0.69
SC	0.71	0.64	0.96	0. 79	0.93	0.76
SC+Prof	0.75	0.67	0.97	0.81	0.94	0.78
SC+Prof+QE	0.77	0.85	0.78	0.78	0.95	0.73
SC+Prof+DE	0.78	0.82	0.83	0.79	0.92	0.77
AC	0.67	0.61	0.89	0. 78	0.93	0.75
AC+Prof	0.71	0.65	0.91	0.81	0.94	0.77
AC+Prof+QE	0.73	0.66	0.93	0.82	0.95	0.78
AC+Prof+DE	0.73	0.67	0.91	0.80	0.92	0.77

The table shows that the complete ENRICH system, which exploits context information, clustering, profiling, and tuple expansion, achieves the highest accuracy, improving F1 measure (compared to the baseline **SharedMatch**) by 28-42% over both domains.

AllAttrMatch, which exploits disjoint attributes, performs better than **S**haredMatch, which only exploits the shared attribute "name", suggesting that the use of extra context attributes is indeed helpful. The F1 measure increases from 0.61 to 0.65 in DBLP and from 0.56 to 0.75 in IMDB. In subsequent discussion, we will use **A**llAttrMatch as our baseline method.

Both clustering algorithms significantly improve F1 on both data sets, compared to **AllAttrMatch**. In particular, the improvement of single link clustering on DBLP is across all three measures (P, R, and $F1$). The results suggest that global information obtained by clustering can be leveraged for better retrieval accuracy.

With the help of profilers, the F1 measure of single-link increases from 0.71 to 0.75 in DBLP and from 0.79 to 0.81 in IMDB. We obtain the similar result for average-link and baseline. Thus, applying profiles improves *every performance measure* on *both data sets* in all the cases.

Finally, the results show that applying tuple expansion improves F1 in most cases. In particular, both expansion techniques improve the F1 measure on DBLP. However, the improvement of query expansion is smaller than that of data expansion for single link, and is larger for average link. It is expected that data expansion could not improve the performance of average link significantly, since the method of computing the similarity of two clusters by average-link is similar to the data expansion idea in some sense.

4.2. Lesion Studies

Table 2 shows the contribution of each ENRICH component (i.e. clustering, profiling, and tuple expansion) to the overall performance. For both data sets, we remove one system component at each time. The table shows that each system component contributes to the overall performance, and that there is no clearly dominant component.

Table 2. Lesion Study

Method	DBLP			IMDB		
	F1	Prec	Rec	F1	Prec	Rec
remove clustering, use QE	0.72	0.64	0.93	0.75	0.96	0.68
remove clustering, use DE	0.77	0.82	0.82	0.79	0.78	0.91
remove profiler, use SC & QE	0.74	0.79	0.81	0.80	0.92	0.79
remove profiler, use AC & QE	0.69	0.62	0.92	0.81	0.95	0.77
remove profiler, use SC & DE	0.75	0.79	0.81	0.80	0.93	0.78
remove profiler, use AC & DE	0.70	0.63	0.90	0.79	0.89	0.79
remove DE/QE, use SC	0.75	0.67	0.97	0.81	0.94	0.78
remove DE/QE, use AC	0.71	0.65	0.91	0.81	0.94	0.77

4.3. Discussions on Clustering Methods

While the average performance is improved with clustering, a closer examination of the results indicates that clustering most likely helps the cases when the seed tuples are far away from the centroid of entity clusters, and may not help when the seed tuples are close to a centroid.

To see why, consider the two scenarios in Figure 8. For one similarity function, if the similarity between any two tuples referring to the same entity is higher than the similarity between any two tuples referring to the different entities, then such a similarity function is in a sense perfect. In this case, both the baseline and clustering method can always achieve the optimal performance with an appropriate similarity threshold. However, with an imperfect similarity function, the performance of the baseline methods can be expected to be much better if the seed tuple is close to a

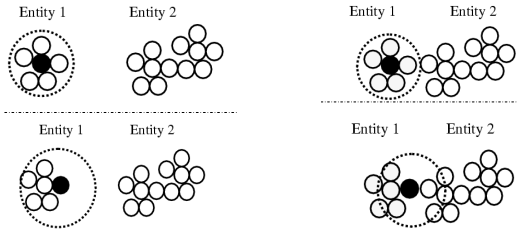


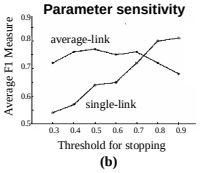
Figure 8. Baseline behavior for perfect (left) and imperfect (right) similarity functions

centroid, but much worse otherwise, which is exactly when clustering may help.

Effect of clustering on boundary and centroid tuples

Method	Centroid Seed			Boundary Seed		
	F1	Prec	Rec	F1	Prec	Rec
Baseline (BL)	0.82	0.86	0.83	0.54	0.51	0.83
SLink	0.79	0.78	0.87	0.67	0.77	0.73
ALink	0.79	0.78	0.88	0.66	0.77	0.71

(a)



(b)

Figure 9. Analysis of clustering methods

To test this hypothesis, we construct two test sets with seed tuples close to a cluster centroid and far from any centroid, respectively, and compare the clustering algorithms with the baseline method on each of the sets. The results on DBLP are shown in the Figure 9(a). We see that clustering indeed performs much better on the set with seed tuples far from a centroid. The results on IMDB are not shown, but are similar.

Comparison: We explore single-link and average-link clustering methods in our experiments. These methods define the similarity between two clusters differently. Single-link computes it as the maximum similarity, whereas average-link computes it as the average similarity over all pairs of tuples in the two clusters.

Unlike average-link, single-link also captures the idea of transitivity, such as if tuple A is similar to tuple B and B is similar to tuple C, then tuple A is similar to tuple C. For example, in Figure 6, we know that paper (1) is the "seed paper". Without the idea of transitivity, paper (3) is not returned in the result, due to the mismatch of the coauthors between paper (1) and paper (3). However, it is clear that paper(2) is similar to both paper(1) and paper(3). By exploiting the idea of transitivity, paper (1) is also similar to paper (3). Thus, both paper(2) and paper (3) are returned as result. In Table 1, the results show that single-link performs better than average-link, which suggests that transitivity can helps improve ER accuracy.

Despite of better retrieval accuracy, single-link is more sensitive to the threshold for stopping, as shown in Figure 9(b). The stable performance of average-link is expected, since every decision is made according to the aggregate property of the clusters instead of effects of a single tuple in the cluster as is the case for single-link.

4.4. Sensitivity Analysis

In both clustering methods, a threshold needs to be learned from the training set and is used to decide when to stop. Figure 9(b) shows that single-link is more sensitive to the threshold than average-link.

We now study how robust the profilers are with respect to its threshold. Figure 10 shows the improvement of using profiles in precision and F1 across a broad range of thresholds on DBLP. (The figure is similar for IMDB data set.) We can see that the profiles are indeed quite robust. They improve precision and F1 across all the threshold values.

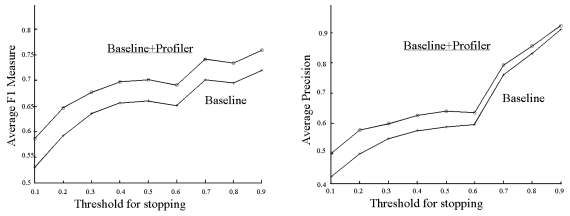


Figure 10. F1 (left) and Precision (right) with and without profiler

Data expansion also needs to learn a threshold from the training data. It is used to decided whether we could expand a tuple with another tuple. When the similarity of two tuples is above the threshold, the two tuples can be merged. Figure 11 shows that the performance on DBLP is sensitive to this threshold, which is intuitively right because augmenting tuples with too much noise should decrease performance. (The figure is similar for IMDB data set.) The figure shows the optimal threshold to be around 0.75. The threshold learned from training data is 0.8, which is close to the optimal threshold.

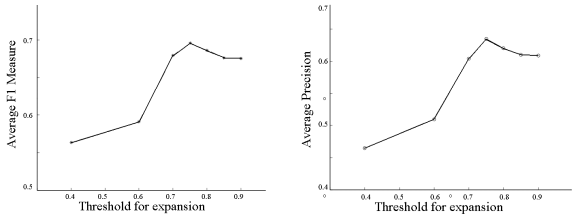


Figure 11. Parameter sensitivity of tuple expansion method: F1(Left) and Precision(right)

4.5. Applying ENRICH to Detect Problematic DBLP Homepages

As a practical application of our ENRICH system we try to clean up DBLP for database researchers in the US. Cur-

rently, there are 290 different database researchers in our system. The cleaned up publication lists are available on the web at hanoi.cs.uiuc.edu/dblife/dblp1/. We compute the jaccard similarity for each researcher based on the two version of the publications lists and we rank the researcher in the increasing order of jaccard similarity. Our hope is that the most problematic pages are ranked at the top.

There are several interesting observations based on the comparison. First, some information at the DBLP site is wrong and our system is able to repair it. Our system can merge the authors from different pages on DBLP. E.g., in DBLP, David Grossman is mentioned on two different pages one as David Grossman and one as David A. Grossman, which are successfully merged by our system. The DBLP lists of Alon Halevy and JungHwan Ho have the same problem. In addition, our system can also split the publications in one page belonging to different researchers(e.g. Chen Li). Second, the lists generated by our system also give us new insights into the work of researchers. Sometimes researchers make a pronounced shift in the field of their work. Our algorithm which assumes continuity in a single field, breaks at this point. This is not always a bad thing, because it shows us changes in a researchers research agenda at a particular time(e.g. William M. Pottenger, Charles M. Eastman and Christopher R. Palmer). Finally, there are also cases that the system goes wrong, we will discuss the limitations of our system in the next subsection.

4.6. Discussion

Effectiveness: We consider the limitations of the current ENRICH system. There are several reasons that prevent ENRICH from reaching 100% precision and recall. First, many times the information containing in a tuple is inherently insufficient to decide to which entity the tuple is related, even for human. We believe exploiting additional information from other data sources could help this case. Second, even with sufficient information, in many cases it is difficult to learn a good similarity function, thereby resulting in less than optimal accuracy. Finally, certain components of the current similarity function are crafted manually, such as the similarity between conferences. Such manual construction at a large scale is error prone, and led to decreased performance. In the future, we plan to employ probabilistic clustering to learn the similarity between conferences automatically from the data.

Efficiency: We now consider the time complexity of ENRICH. Let N be the number of tuples in the data set, and M be the number of tuples in the returned result. We assume the computation time of similarity between two tuples/clusters is constant. In pairwise matching methods, each tuple needs to be compared with the "seed tuple", resulting in time complexity $O(N)$.

In global matching methods, we keep merging two most similar clusters/tuples until the similarity between the two most similar clusters/tuples is below a threshold. In each iteration, we need to choose two tuples from n ($n \leq N$) tuples, so the time complexity is $O(N^2)$. After each iteration, the number of tuples decreases by 1, so the number of iterations is at most N. Therefore, the worst case for global matching method is $O(N^3)$.

For profiling algorithms, we use greedy search to remove the tuple so that the remaining tuples in the result satisfy the profiler. It takes $O(M)$ time to find each such tuple. Since we at most remove $M-1$ tuples, the time complexity for profiling algorithm is $O(M^2)$.

In query expansion, we need to find the most similar tuple for the seed tuple and merge it with the seed tuple. So the time complexity of query expansion is $O(N)$. In tuple expansion, we need to find the most similar tuple for each tuple. It is trivial that the complexity of tuple expansion is $O(N^2)$. Note that tuple expansion is query independent and thus can be implemented offline. In the long term, scaling up ER to very large data sets is another important direction that we plan to pursue.

5. Related Work

The work [20] sketches the ER problem, with no formal problem definitions nor systematic solutions. Several recent works [12, 3, 25] also address issues related to ER, but in the Semantic Web context. The work [12] also matches entities across disparate sources, but uses shared keys and a bootstrapping approach. As far as we can tell, no work has considered profiling over aggregate properties of the gathered information for ER purpose, as we do here. Our work also bears some resemblances to [17]. This work mines all information about a topic (e.g., "data mining") from the Web. In contrast, we mine information about a real-world entity from structured data.

Many works refer to "entity/object consolidation" and "integration", but actually discuss entity matching. Entity matching has received much attention in the database, AI, and data mining communities (e.g. [26, 7, 18, 27, 4, 16, 2, 24, 11, 13, 10, 22, 9]). Earlier solutions employ manually specified rules [13], while many subsequent ones learn matching rules from training data [26, 4, 24]. Several solutions focus on efficient techniques to match strings [19, 11]. Others also address techniques to scale up to very large number of tuples [18, 8]. All above solutions (with the possible exception of [2]) match entities by comparing shared attributes. Our solution innovates by utilizing also the disjoint attributes (e.g., in the **AllAttrMatch** method), to maximize ER accuracy.

Clustering techniques have been employed in entity matching (e.g., to bring potentially similar tuples together [18, 8]), but not in ER. Entity matching finds *all*

matching tuple pairs, which typically belong to *multiple* real-world entities. In contrast, ER finds all tuples that belong to just a *single* real-world entity. As such, it appears that pairwise matching is sufficient for ER. However, a key contribution of our work here is to show that clustering helps to obtain a "global view", which significantly improves entity retrieval performance. ER can be considered a more general problem than entity matching, in that a solution to ER is applicable also to the entity matching contexts.

6. Conclusion & Future Work

In this paper we have developed a novel solution to the ER problem, which plays a fundamental role in numerous information management applications, but has received little attention.

We focused on ER over structured data. In this context, we showed that even though any entity matching solution can be applied in a straightforward, pairwise manner to ER, a better solution can be developed. This solution in essence employs clustering to obtain a "global view", then leverages the "view" to make better matching decisions. We further examined adding profilers and tuple expansion, and showed that the combination of all these techniques achieves high accuracy on the ER problem. The extensive experiments on real-world data clearly showed the utility of our approaches.

We plan to extend our current work in several ways. First, we will examine more expressive ER variations, moving beyond structured data to text and the Web. Second, we plan to develop a probability-based formalization of our solutions, using generative models. Such a formal framework will allow us to study the problem and our solutions in depth, and help better compare them with other works.

References

[1] http://www.informatik.uni-trier.de/ ley/db/.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proc. of VLDB-02*.

[3] K. Anyanwu and A. Sheth. ρ-queries: Enabling querying for semantic associations on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-03)*, 2003.

[4] M. Bilenko and R. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX, Feb. 2002.

[5] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name-matching in information integration. In *IEEE Intelligen Systems*, 2003.

[6] C. Buckley, J. Allan, G. Salton, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Proc. of the Third Text REtrieval Conference(TREC-3)*, 1995.

[7] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Procceedings of SIGMOD-98*, 1998.

[8] W. Cohen and J. Richman. Learning to match and cluster entity names. In *Proc. of SIGKDD-02*.

[9] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley and Sons, 2003.

[10] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. In *Proc. of ICDE-00*.

[11] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text join for data cleansing and integration in an rdbms. In *Proc. of ICDE-03*.

[12] R. Guha. Object co-identification on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-04)*, 2004.

[13] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.

[14] D. W. Hosmer and S. Lemeshow. *Applied logistic regression*. John Wiley & Sons, 1989.

[15] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[16] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proc. of the 3rd Int. Conf. on Autonomous Agents*, 1999.

[17] B. Liu, C. Chin, and H. NG. Mining topic-specific concepts and definitions on the web. In *Proceedings of the World-Wide Web Conference (WWW-03)*, 2003.

[18] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of SIGKDD-00*.

[19] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996.

[20] R. Nado and S. Huffman. Extracting entity profiles from semistructured information spaces. In *SIGMOD Record 26(4):32-38*, 1997.

[21] E. Rahm and P. Bernstein. On matching schemas automatically. *VLDB Journal*, 10(4), 2001.

[22] V. Raman and J. Hellerstein. Potter's wheel: An interactive data cleaning system. In *The VLDB Journal*, pages 381–390, 2001.

[23] J. J. Rocchio. Relevance feedback in information retrieval. *The SMART Retrieval System-Experiments in Automatic Document Processing*, pages 313–323, 1973.

[24] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. of SIGKDD-02*.

[25] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. Managing semantic content for the web. *IEEE Internet Computing*, pages 80–87, 2002.

[26] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of the 8th SIGKDD Int. Conf. (KDD-2002)*, 2002.

[27] W. Yih and D. Roth. Probabilistic reasoning for entity and relation recognition. In *Proc. of COLING'02*, 2002.